Лекция 8

Работа с файлами

Базовые определения

Файл — именованная (или адресуемая иным способом) область данных на носителе информации, используемая как базовый объект взаимодействия с данными в операционных системах.

Файловая система — порядок, определяющий способ организации, хранения и именования данных на носителях информации. Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов (и каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла.

Работа с файловой системой

Работа с элементами файловой системы основана на вызове функций операционной системы. Таким образом методы и классы реализующие в Java работу с файловой системой вызывают функции операционной системы (используются native методы).

В Java существует несколько классов для работы с файловой системой. Одним из самых популярных является класс File (для его работы нужно импортировать пакет java.io.File). Объект этого класса представляет собой адрес объекта файловой системы.

Для создания объекта типа файл используется адрес (абсолютный или относительный) объекта файловой системы заданный в виде строки.

Абсолютный адрес — адрес заданный от верха иерархии файловой системы (от точки монтирования в Linux, MacOS или логического диска в Windows).

Относительный адрес — адрес указанный относительно объекта файловой системы (по умолчанию относительно рабочего каталога приложения).

Относительный адрес

```
File file1 = new File("a.txt");
```

Методы доступные в классе File

Метод	Описание
String getAbsolutePath()	Вернет абсолютный адрес в виде строки
String getName()	Вернет имя объекта файловой системы
boolean isDirectory()	Вернет true если адресуется каталог
boolean isFile()	Вернет true если адресуется файл
boolean exists()	Вернет true если адресуется существующий объект файловой системы
long length()	Вернет размер объекта файловой системы в байтах
File[] listFiles()	Если адресуется каталог то вернет массив адресов содержимого каталога
boolean createNewFile()	Создание нового файла. Вернет true если файл удалось создать
boolean mkdirs()	Создает новый каталог. Вернет true если каталог удалось создать
boolean renameTo(File dest)	Перенос объекта файловой системы. Вернет true если перенос успешен
boolean delete()	Удаляет объект файловой системы. Вернет true в случае успешности операции

И еще много методов. Более подробно можно прочесть тут - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/io/File.html

Пример создания файла

```
File file1 = new File("a.txt");

try {
    file1.createNewFile();
} catch (IOException e) {
    e.printStackTrace();
}

File workFolder = new File(".");
File[] files = workFolder.listFiles();
for (int i = 0; i < files.length; i++) {
    System.out.println(files[i]);
}
```

Указание адреса где нужно создать файл

Создание файла по указанному адресу

Получение адреса рабочего каталога

Получение массива адресов объектов в рабочем каталоге

Обработчик исключений try - catch

Блок контролируемого кода вкладывается в блок try. Блок try определяет локальную область видимости (поэтому все что объявлено в нем не доступно за его пределами).

Обработчик исключений реализован с помощью синтаксической конструкции catch. Блок характеризуется описанием типа обрабатываемого исключения и блока кода, выполняемого при обработке исключения. Исключение обрабатывается или в случае точного совпадения типа, или в случае, если в блоке catch описан суперкласс обрабатываемого исключения.

В общем случае конструкция выглядит следующим образом:

```
try{
    Контролируемый код
} catch (Exception_type e){
    Код выполняемый при обработке исключения
}
```

Пример обработчика try - catch

```
import java.io.File;
import java.io.IOException;

public class Main {

    public static void main(String[] args) {

        File file1 = new File("a.txt");

        try {

            file1.createNewFile();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Контролируемый код

Обработчик catch

Создание каталога

```
File folder1 = new File("AAAA");
boolean result = folder1.mkdirs();
System.out.println("folder create " + result);
```

Указание адреса где нужно создать каталога

Создание каталога по указанному адресу

Обратите внимание, что в случае неудачи при создании каталога исключение не создается. Если каталог не создается, то результатом работы метода будет значение false.

Указание адреса на основе другого адреса

Довольно часто нужно указать относительный адрес одного объекта файловой системы относительно другого (например, для упрощения создания файла в определенном каталоге). Для этого при создании объекта класса File используется два параметра. Первый адрес существующего объекта файловой системы (в виде объекта класса File), второй относительный адрес (задается строкой).

```
File folder1 = new File("AAAA");

boolean result = folder1.mkdirs();

File file2 = new File(folder1, "abc.doc");

try {
    file2.createNewFile();
} catch (IOException e) {
    e.printStackTrace();
}
```

Указание адреса относительно существующего каталога

Удаление объектов файловой системы

Для удаления объекта файловой системы стоит указать адрес удаляемого объекта и использовать метод delete.

```
File file1 = new File("a.txt");

file1.delete();
```

Удаление файла по указанному адресу

Внимание! Для удаления каталога сначала нужно удалить все его содержимое. В противном случае каталог не будет удален.

Запись данных в символьном виде в файл

Для записи данных в символьном виде (в виде последовательности строк) можно использовать PrintWriter. Для его работы необходимо импортировать пакет java.io.PrintWriter. При его создании нужно указать в какой файл производить запись данных. Для этого адрес файла указывается или в виде объекта типа File или в виде строки. Если файл с указанным адресом не существует, то он автоматически будет создан. Создавать и работать с PrintWriter нужно в блоке try-with-resources. Этот блок автоматически закрывает все открытые потоки ввода-вывода. Запись производиться с помощью вызовов методов print() и println().

Важно помнить что PrintWriter по умолчанию переписывает данные в файле. Т.е. если указать для записи уже существующий файл то все данные в нем будут удалены и запись будет производится в пустой файл.

Пример применения PrintWriter

```
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;

public class Main {

    public static void main(String[] args) {

        File file1 = new File("hello.txt");

        try (PrintWriter pw = new PrintWriter(file1)) {
            pw.println("Hello world");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Адрес файла для записи

Создание PrintWriter

Запись строки в файл

Вычитка данных из файла с помощью Scanner

С помощью Scanner можно вычитывать данные с символьном представлении из файла. Для этого в качестве источника данных нужно указать адрес файла (в виде объекта типа File). Открывать файл на чтение с помощью Scanner нужно в блоке try-with-resources.

Для определения есть ли в файле еще строки стоит использовать метод hasNextLine(). Этот метод вернет true если в файле еще есть строки для вычитки и false в противном случае.

Пример метода для вычитки содержимого файла в виде строки

```
public static String getStringFromFile(File file) {
    String result = "";
    try (Scanner sc = new Scanner(file)) {

        for (; sc.hasNextLine();) {
            result += sc.nextLine() + System.lineSeparator();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return result.substring(0, result.length() - 1);
}
```

Scanner для вычитки из файла

Проверка наличия строки для вычитки

Вычитка строки

Задание для самостоятельной проработки.

1) Создайте консольный «текстовый редактор» с возможностью сохранения набранного текста в файл.
2) Напишите метод для сохранения в текстовый файл двухмерного массива целых чисел.
3) Реализуйте метод который выведет на экран список всех каталогов расположенных в каталоге адрес которого будет параметром этого метода.

Дополнительное задание для самостоятельной проработки

1) Считайте из текстового файла текст на английском языке и выведите статистику по частоте использования букв в тексте (т. е. буква — количество использования), причем первыми должны выводиться буквы используемые чаще всего.
2) Напишите метод для создания в файле ASCII — арта, т. е. фигуры размером примерно 40x40 символов заполненных символами образующими узор.

Список литературы

1) Герберт Шилд — Java 8. Полное руководство. 9-е издание, «Вильямс» — 2015
2) Кей Хорстман, Гари Корнел — Библиотека профессионала Java. Том 1. Основы. 9-е издание, «Вильямс» — 2014